IBM Informix® Dynamic Server™ (IDS)
IDS Problem Determination Tutorial Series

Enterprise Replication (ER) Problem Determination

Table of Contents

Enterprise Replication (ER) Problem Determination

About this tutorial

Introduction

This tutorial assists you in diagnosing common problems with Enterprise Replication (ER). The focus of this tutorial is to teach you skills for determining common problems related to ER.

Audience and assumptions

Prior to taking this tutorial, you should be familiar with the functionality of the IDS product and comfortable executing IDS commands and SQL statements. This includes a familiarity with Administration of IDS (onmode and onspaces), Data Migration tools (load/unload), object creation, and data access.

Tutorial objectives

With this tutorial you can develop the skills to solve basic problems which can occur with Enterprise Replication. Each section covers an area known to cause problems with customers using ER for the first time.

This tutorial explains the replication model, monitoring, and determination of Enterprise Replication problems, by way of hands-on exercises, review on ER and Engine command output, and review of log files.

Setup

This tutorial's examples use IDS 9.40.UC1 installed on the UNIX® operating system, but most concepts and exercises can be completed on any IDS system.

To work through the examples in this tutorial, you should have completed the following tasks:

- Installed 2 instances of IDS on 1 or 2 systems.
- Initialized each instance with 1 root dbspace and 1 smart blob space named 'sbspace'.
- Created 1 database named 'ertest' with unbuffered logging on each instance.

Tutorial Conventions Used

When a tool or utility is first mentioned, it will be in shown in **bold** text.

All commands, statements, and their output will be shown in `monospaced` font.

In general, examples shown will be UNIX® based. However, they should also work in the Windows® environment (unless otherwise noted).

Some of the examples in this tutorial show supported options for the utilities. These options may change from one release to the next, but the basics should always remain the same.

About the author

Ron Privett joined IBM as part of the Informix acquisition. He started with Informix Technical Support in 1997. After four years as a Frontline engineer, and teaching other engineers how to support Enterprise Replication, Ron is currently a member of the IDS Down Systems and Diagnostics Group. His main responsibilities on this team include working on IDS down systems and specializing in Replication issues.

You can reach Ron by locating his e-mail address in the IBM Global Directory at http://www.ibm.com/contact/employees/us .

Section 1 Troubleshooting Configuration Problems

Section 1.1 Introduction

Just like IDS, the decisions you make before you run oninit –i (the command to initialize the instance from scratch) for the first time can determine how your server will operate under certain conditions. When troubleshooting configuration issues, you need a clear understanding of ER's requirements, limitations, connectivity, and administration.

Before starting ER, consider performing the following tasks: adding a smart blob space, altering tables, providing file system space for log files, network bandwidth, and scripting issues for re-defining ER.

Section 1.2 Requirements and Limitations

Before starting to determine configuration problems, be sure ER's basic requirements have been met.

Requirement	Description
Logging	This is required as ER is a log reading application. Any form of logging can be used: ANSI, buffered, unbuffered.
Trusted Connectivity	All root servers must be able to communicate with each other via TCP/IP. Both servers must be trusted to participate in replication. Use *`dbaccess -> connection -> connect`*, without a username/password, to insure a trusted connection. (NOTE: In 9.40.UC2, the entire system does not need to be fully trusted. This version introduced a method to specify the trust relationship for only the ER ports. For more information about this feature, please refer to your Release Notes.)
Primary Key	All replicated tables must have a primary key constraint.
Server Time	All ER servers need synchronized clocks. ER uses GMT internally. If your ER servers are in different Time Zones, insure that each server's time is set for your Local Time Zone.

Also, make sure the following limitations have not been exceeded.

Limitation	Description
ER and HDR	ER and HDR on the same engine are only supported when using version 9.40 of the engine.
No Views, Aggregates, or Joins	ER only works with base tables and cannot derive data from view or joins.
Cannot Alter Replicated	If a replicate is defined for a table, ER prevents it from being altered. This is because tables currently being altered insert new

Tables	version pages of the table in the logical log. ER expects log pages for replicated tables to have the same format as it did when the replicate was defined, so any logical log page in a new format will not be snooped correctly.

Section 1.3 Connectivity and the SQLHOSTS file

Connectivity Issues:

Networks, servers, and security protocols change. To manage anticipated changes, you will need to determine when connectivity between ER servers has been lost. When checking a connectivity issue, be sure to confirm the following:

- Test for 'Trusted Connectivity' between all problem servers using the ***`dbaccess -> connection -> connect`***. On UNIX, there are several ways to trust systems, for example: 1) adding entries to the '.rhosts' file in the Informix account's home directory, 2) adding a /etc/hosts.equiv file. On Windows platforms, check the %System Dir%\system32\drivers\etc directory for the hosts.equiv file. (NOTE: This method will only work if you are NOT using the added security features to only trust ER ports.
- In DNS environments, use 'nslookup' to ensure the hostname and IP can be found.
- In NIS environments, use 'ypcat hosts | grep *hostname or ipaddress*'

SQLHOSTS file Issues:

Capture the SQLHOSTS file for all servers experiencing connectivity problems. The following is an example of what this file should look like if you have four active ER servers:

```
dbserver    nettype     host              service options
g_bono      group       -                 -       i=1
bono        ontlitcp    dublin.com        5532    g=g_bono

g_edge      group       -                 -       i=2
edge        ontlitcp    london.com        5533    g=g_edge

g_larry     group       -                 -       i=3
larry       ontlitcp    artane.com        5234    g=g_larry

g_adam      group       -                 -       i=4
adam        ontlitcp    oxfordshire.com   5235    g=g_adam
```

Note the following guidelines when checking the SQLHOSTS file:

- In an ER only environment, there is a one to one relationship between the group name and its dbserver. For example, you cannot have multiple dbservers as a subset under one group name. In an ER/HDR environment, the group can have multiple dbservers, as long as the dbservers are grouped together into a logical unit.
- Shared memory connections should not be contained in a group used by ER. ER only works over TCP network connections.
- The group name, dbservername, and site identifier (**i=**n) must each be unique. The site identifier must be an integer value.

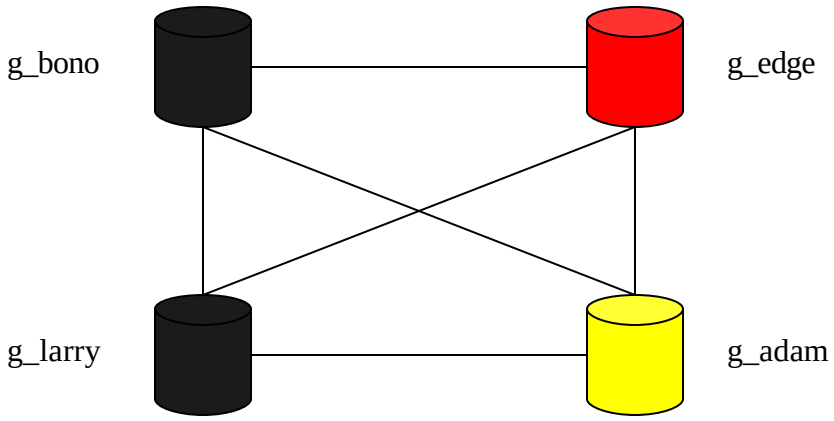
- The ‘g’ option must be specified for all dbservernames and aliases. This option points back to the group name for that specific instance.
- Entries in this file should be consistent across all ER servers.
- The items marked in **bold and red** are not optional or changeable.

Section 1.4 Defining Servers

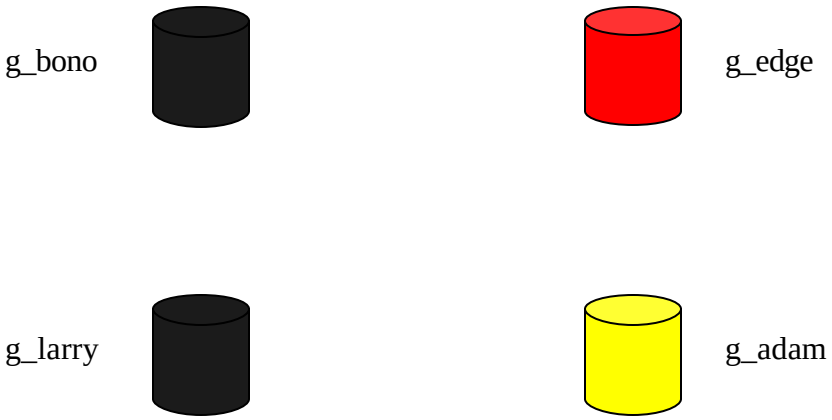
When defining ER servers, the majority of problems can be avoided by using caution. Using the SQLHOSTS file from Section 1.3, here is one method to define the servers, running each command from its own server:

```
dublin.com> cdr define server –I g_bono
london.com> cdr define server –I g_edge –S g_bono
artane.com> cdr define server –I g_larry –S g_bono
oxfordshire.com> cdr define server –I g_adam –S g_bono
```

This will result in the following ‘fully connected’ ER configuration:



One problem many DBA’s experience is leaving off the ‘-S’ option to sync this server with another. If this option is not used, ER will start, but the individual servers will not be connected to each other. This will result in the following ‘standalone’ ER configuration:



Also, time on each ER server should be synchronized. ER uses GMT time internally, the same time system UNIX systems use. If you have servers in different time zones, be sure each server's time is synchronized 'within' its own local time zone.

Section 1.5 Analyzing Connection Data

The following onstats and cdr commands will provide useful information about the status of the ER connections:

onstat –g cat:

```
Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 19:19:32 --
28672 Kbytes

SERVERS
-------------------
  Current server  : Id 8, Nm g_edge
  Last server slot: (0, 2)
  # free slots    : 0
  Broadcast map   : <[0005]>
  Leaf server map : <[0000]>
  Root server map : <[0006]>
  Adjacent server map: <[0004]>
    Id: 08, Nm: g_edge, Or: 0x0002, off: 0, idle: 0, state Active
      root Id: 00, forward Id: 00, ishub: FALSE, isleaf: FALSE
      subtree map: <empty>
        NifInfo: b7446b0
                 Last Updated    (0) 1969/12/31 18:00:00
                 State           Local server
                 Total sent      0 (0 bytes)
                 Total recv'd    0 (0 bytes)
                 Retry           0 (0 attempts)
                 Connected       0

    Id: 09, Nm: g_bono, Or: 0x0004, off: 0, idle: 0, state Active
      root Id: 00, forward Id: 09, ishub: FALSE, isleaf: FALSE
      subtree map: <empty>
        NifInfo: b744730
```

```
Last Updated    (1053093691) 2003/05/16 09:01:31
State           Connected
Total sent      41 (1550 bytes)
Total recv'd    6 (2320 bytes)
Retry           0 (0 attempts)
Connected       1
```

When checking for problems in the connection – look at the 'State' of the connection. The possible values for the 'State' field are: Never Connected, Disconnected, Timeout, Logic error, Start error, Admin close, Connected, Connecting, and Local Server. A 'local server' is the server where the onstat –g cat command was run, and a 'connected' server is a server is currently connected to the local server and functioning normally.

onstat –g nif:

```
Informix Dynamic Server Version 9.40.UC1     -- On-Line -- Up 04:25:16 --
28672 Kbytes

NIF anchor Block: b627470
        nifGState               RUN
           RetryTimeout         300

CDR connections:
 Id    Name               State                Version      Sent  Received
-------------------------------------------------------------------------
   8 g_edge               RUN                        7       271        10
```

The 'State' column can have the following values: INIT, INTR, ABORT, SHUT, SLEEP, RUN, STOP, TIMEOUT, BLOCK, SUSPEND. If there is any other state than 'RUN' then ER is inactive at that site for the reason listed.

cdr list server:

```
SERVER                  ID STATE    STATUS     QUEUE  CONNECTION CHANGED
-------------------------------------------------------------------------
g_edge                   8 Active   Connected       0 May 16 09:01:32
g_bono                   9 Active   Local           0
```

Use this information when trying to determine how the other servers see an ER server that may have problems. For example, if the send queues of Servers A and B are filling up with data for Server C, then check the cdr list server from both A and B to see what status they show for Server C. Then, compare this with the 'cdr list server' command from Server C.

Also, the online.log contains important information about connections of other ER servers. Review this file for any errors or messages.

Section 1.6 Connectivity Exercise

Using the IDS engines that you started with in Tutorial Setup section, do the following:

1. Set up both SQLHOSTS files based on the following example, making changes for your specific server names, host names, SOC or TCP protocol, and group names:

1st Server:

```
dbserver    nettype     host         service     options
g_bono      group       -            -           i=1
bono        ontlitcp    dublin.com   5532        g=g_bono

g_edge      group       -            -           i=2
edge        ontlitcp    london.com   5533        g=g_edge
```

2nd Server:

```
dbserver    nettype     host         service     options
g_edge      group       -            -           i=2
edge        ontlitcp    london.com   5533        g=g_edge

g_bono      group       -            -           i=1
bono        ontlitcp    dublin.com   5532        g=g_bono
```

2. Define 2 ER servers that are synchronized with each other. The following commands are an example:

1st Server:

```
cdr define server –A {dir} –R {dir} –I g_bono
```

2nd Server:

```
cdr define server –A {dir} –R {dir} –S g_bono –I g_edge
```

3. Both ER servers should now be running, and able to see the other. The cdr list server command should show the following:

1st Server:

```
SERVER             ID STATE    STATUS     QUEUE  CONNECTION CHANGED
-------------------------------------------------------------------
g_edge              8 Active   Connected      0 Jul  5 17:45:51
g_bono              9 Active   Local          0
```

2nd Server:

```
SERVER            ID STATE    STATUS     QUEUE  CONNECTION CHANGED
-------------------------------------------------------------------
g_edge              8 Active   Local          0
g_bono              9 Active   Connected      0 Jul  5 17:45:53
```

4. Delete both servers with the following command:

1st Server:

```
cdr delete server g_bono
```

2nd Server:

```
cdr delete server g_edge
```

5. Change the SQLHOSTS file on the 1st server – to not have any ‘group’ syntax. For example:

1st Server:

```
dbserver    nettype     host         service     options
bono        ontlitcp    dublin.com   5532
edge        ontlitcp    london.com   5533
```

6. Run the command to define the 1st ER server (see Step 2). Did you receive the following error?

```
command failed -- server not defined in sqlhosts  (35)
```

. . . and see this in the online.log file:

```
08:18:32  Building 'syscdr' database ...
08:18:36  'syscdr' database built successfully.
08:18:37  CDR current servername not found
```

7. Correct the problem with the SQLHOSTS file, and define the server correctly. Also, create directories to hold ATS and RIS files, then use these directories when defining the servers.

Another exercise to show the effects of a down server:

1. Bring one ER server down with onmode –ky.
2. On the other server, check the status of the down server using some of the onstat and cdr commands from Section 1.5. For example:

```
cdr list server
SERVER               ID STATE    STATUS     QUEUE  CONNECTION CHANGED
-------------------------------------------------------------------------
g_edge                8 Active   Local          0
g_bono                9 Active   Dropped        0 Jul  8 08:40:34

onstat -m
08:40:35  CDR NIF receive failed asfcode: -25582 oserr 9
08:40:35  CDR connection to server lost, id 9, name <g_bono> Reason:
connection lost

onstat -g nif
NIF anchor Block: b75f470
        nifGState              RUN
           RetryTimeout        300

CDR connections:
 Id    Name              State           Version       Sent   Received
-------------------------------------------------------------------------

onstat -g ath|grep CDR
 40      b6c9018  adde098  2    sleeping secs: 4       3cpu       CDRSchedMgr
 42      b6c9418  adde6a8  2    sleeping secs: 99      5cpu       CDRDTCleaner
 43      b6c9a80  addecb8  2    cond wait  CDRCparse   1cpu       CDRCparse
 47      b741e58  addce68  2    sleeping secs: 1       3cpu       CDRM_Monitor
 48      b53fda8  addf2c8  2    cond wait  CDRAckslp   4cpu       CDRACK_0
 49      b78f018  addf8d8  2    cond wait  CDRAckslp   1cpu       CDRACK_1
```

```
 50      b78f160  addfee8  2    cond wait  CDRDssleep   3cpu        CDRD_1
 51      b78f2e0  ade04f8  2    cond wait  CDRDssleep   4cpu        CDRD_2
 52      b78f460  ade0b08  2    cond wait  CDRDssleep   3cpu        CDRD_3
 53      b78f5e0  ade1118  2    cond wait  CDRDssleep   4cpu        CDRD_4
```

3. How can you tell this server is down, from the running ER server?

   Check the status of the server in 'cdr list server':

```
cdr list server
SERVER               ID STATE    STATUS     QUEUE  CONNECTION CHANGED
------------------------------------------------------------------------
g_edge                8 Active   Local          0
g_bono                9 Active   Dropped        0 Jul  8 08:40:34
```

   Check the online log:

```
onstat -m
08:40:35  CDR NIF receive failed asfcode: -25582 oserr 9
08:40:35  CDR connection to server lost, id 9, name <g_bono> Reason:
connection lost
```

   Check the Network Interface via onstat:

```
onstat -g nif
NIF anchor Block: b75f470
        nifGState               RUN
           RetryTimeout         300

CDR connections:
 Id    Name              State           Version       Sent   Received
------------------------------------------------------------------------
  {no sites listed here}
```

   Check the ER threads running on this engine:

```
onstat -g ath|grep CDR
 77      b302368  addc858  2   sleeping secs: 49   3cpu        CDRNsA9
 78      b890070  addda88  2   cond wait  netnorm  1cpu        CDRNrA9
```

   When both sites are connected you will see 2 (or more) threads with the name CDRNxAn. Where the x can be **s** for send, or **r** for receive. And the n is the unique group id for this site as listed in the sqlhosts file.

Section 2 Troubleshooting Replicate Problems

Section 2.1 Introduction

Replicates are the building blocks of ER. They define what data gets replicated, as well as where it is replicated to and when. When troubleshooting issues with individual replicates, keep in mind table level requirements, replicate requirements, and how the replicate was defined.

Section 2.2 Table Requirements

Base tables must have the following properties:

- Table cannot be a view, or a synonym to another table.
- Table must have a primary key (PK).
- Table must be in a database with logging.
- PK's must be the same on each table on each ER server.
- Cannot drop or alter a table defined for replication.
- Support of Smart Large Object (BLOB/CLOB), and Opaque user-defined data types (UDTs) columns if Blade supports streamread() and streamwrite() functions.
- Support for Named and unnamed ROW data types in 9.4 only
- Support for Collection data types: LIST, MULTIST, and SET in 9.40 only

Section 2.3 Replicate Requirements

Individual replicates must have the following properties:

- Replicate names must be unique.
- Select used to define replicate cannot contain a view, join condition, aggregate clause, or a sub-query.

Section 2.4 Defining Replicates

Problems with ER may be traced back to a problem with a replicate definition. When defining a replicate, keep in mind the following: transaction level scope, conflict type, ATS and RIS spooling, frequency of replication, and any triggers that fire this replicate.

In addition to the semantics, there can also be 'logic errors' when defining replicates. For example, if the replicate is defined with a 'where' clause, check it to insure duplicate data will not be sent. If one participant modifier includes WHERE x < 50 and another includes WHERE x < 100, Enterprise Replication sends the data twice.

Section 2.5 Analyzing Replicate Data

Remember that any data you get from onstat comes directly from Shared Memory, and any CDR command or select from syscdr comes from disk. There is a chance that these two 'copies' of the state of ER could be different. You can compare the two by running the following sql statements against the syscdr database:

```
> select * from syscdr:repdef where repname='rocketRep';

   repid         524289
   replsetid     0
   repstate      2
   flags         268504320
   repname       rocketRep
   cr_primary    T
   cr_secondary
   cr_spopt
   cr_spname
   freqtype      C
   create_time   1053275103
   modify_time   1053275103
   susp_time     0

   1 row(s) retrieved.

> select * from partdef where repid in
 (select repid from repdef where repname='rocketRep');

   repid       524289
   servid      8
   partnum     1048687
   partstate   2
   partmode    P
   flags       0
   start_time  0
   stop_time   0
   db          test
   owner       informix
   table       rocket
   selecstmt   select * from rocket

   repid       524289
   servid      9
   partnum     0
   partstate   2
   partmode    P
   flags       0
   start_time  0
   stop_time   0
   db          test
   owner       informix
   table       rocket
   selecstmt   select * from rocket

   2 row(s) retrieved.
```

The following onstat –g cat output can be helpful in establishing the specific status of a replicate. Some of the key fields to look at are noted in **bold**..

onstat –g cat:

```
REPLICATES
```

```
  Parsed statements:
          Id 524289 table rocket

  Inuse databases: test(1)
    Name: rocketRep, Id: 524289 State: ACTIVE Flags: 0  use 0 lastexec
Wed Dec 31 18:00:00 1969

        Local Participant: test:informix.rocket
        Attributes: TXN scope, Enable ATS, Enable RIS, all columns sent
in updates
        Conflict resolution: [TIMESTAMP]
        Column Mapping: ON, columns INORDER, offset 8, uncomp_len 38
        No Replicated UDT Columns
        Bitmap: all <[0006]> route <[0004]>
```

Definition of **bold** terms

- Name: The name of the replicate
- Id: A unique number that identifies this replicate
- State: Current state of the replicate could be one of (FAILED, SUSPENDED, CAT_ONLY, INACTIVE, ACTIVE, DELETED, RESUMED)
- Local Participant: Name of database:owner.tablename this replicate is defined for locally.
- Conflict resolution: Lists the type of Conflict resolution used for this replicate. Could be one of the following: TIMESTAMP, STORED PROCEDURE, IGNORE

If a replicate is not sending data – insure it is not a time based replicate using onstat –g rep. Only time based replicates are listed with ‘onstat –g rep’. If the replicate is time based – check the ‘Next’ field for time that the replicate runs.

onstat –g rep:

```
                  Time based replicates

Name                Frequency           Last            Next
=========================================================================
rep_rocket          Every 10 min        Jul 10 11:19:25 Jul 10 11:29:25

Schedule manager Cb: b6aee40 State: 0x8100 <CDRINIT,CDRRUNNING>

Event        Thread             When
=========================================
CDRDS          CDREvent       00:00:20
CDRRepSched    CDRRep         00:08:37
```

In the above onstat –g rep output, you can see one time based replicate named ‘rep_rocket’ that replicates at an interval every 10 minutes. Also, the CDRRep thread counts down to the next time when the replicate will replicate any waiting data.

Another command used to monitor replicates is ‘cdr list repl’ as seen below:

cdr list repl *replicatename:*


```
DEFINED REPLICATES ATTRIBUTES
REPLICATE:     rocketRep
STATE:         Active
CONFLICT:      Timestamp
FREQUENCY:     immediate
QUEUE SIZE:    0
PARTICIPANT:   test:informix.rocket
OPTIONS:       transaction,ris,ats,fullrow
```

Lastly, be sure to check the syscdr:cdr_errors table for any errors which may have occurred with your replicates. You can also run 'cdr error' from the command line. In the following example, a replicate failed to define due to a table definition error:

cdr error:

```
SERVER:SEQNO            REVIEW   TIME                      ERROR
g_bono:1                       N   2003-07-10 13:24:40     18
        GC operation define replicate 'rep_launch' failed: table does not
contain primary key
```

Section 2.6 Defining Replicate Exercise

Using the ER servers you stated in Section 1.6, do the following:

1. Create the following table on both ER servers:

```
CREATE TABLE table_a
  (
  col_1   INT,
  col_2 CHAR(10),
  PRIMARY KEY (col_1)
  ) WITH crcols;
```

2. Using the following example syntax – define an Update-Anywhere replicate:

```
cdr define replicate –Cignore –S tran –R –A rep_table_a \
"ertest@g_bono:informix.table_a" "select * from table_a"\
"ertest@g_edge:informix.table_a" "select * from table_a"
```

3. Start the replicate and insure it is replicating from both servers:

```
cdr start repl rep_table_a g_bono g_edge
```

4. Delete this replicate (use 'cdr list repl' to insure it is removed).

```
cdr delete repl rep_table_a
```

5. Drop the table 'table_a' on only one ER server, then run this sql:

```
CREATE TABLE table_a
  (
  col_1     INT,
```



```
col_2     CHAR(10),
PRIMARY KEY (col_2)
) WITH crcols;
```

6. Use the example syntax from step 2 and define an Update-Anywhere replicate.
7. Did the replicate define?

   Yes, in this case the replicate will define because ER checks for the existence of a primary key – but not that the PK's on both servers are the same.

8. Insert the following rows on the server with col_1 as the primary key:

```
insert into table_a values (1,'a');
insert into table_a values (2,'a');
```

9. Did you see the following error?

```
15:50:25  CDR CDRD_2: transaction aborted (One or more rows in a
transaction defined with tx scope were rejected) with sql error 0 isam
error 0.
15:50:25  CDR CDRD_2: failed transaction spooled to file
/usr/940/ats/ats.g_bono.g_edge.D_2.030710_15:50:25.4
```

10. Can you tell from the following ATS file what happened on the target server?

```
TXH RIS file:/usr/940/ris/ris.g_bono.g_edge.D_2.030710_15:50:25.3 has
also been created for this transaction
==========
TXH Source ID:8 / Name:g_edge / CommitTime:03-07-10 15:50:23
TXH Target ID:9 / Name:g_bono / ReceiveTime:03-07-10 15:50:25
TXH Number of rows processed when transaction was aborted:1
TXH One or more rows in a transaction defined with tx scope were
rejected
TXH CDR:5 (Error: Insert aborted, row already exists in target table) /
SQL:0 / ISAM:0
----------
RRH Row:1 / Replicate Id: 524295 / Table: test@informix.table_a /
DbOp:Insert
RRD 2|a
```

11. Correct the problem with the primary key, and define the replicate correctly using the syntax provided in step 2.

Section 3 Conflict Resolution Problems

Section 3.1 Introduction

Conflict Resolution issues are different than most ER problems in that they are often the easiest ones to determine. The Aborted Transaction Spooling (ATS) and Row Information Spooling (RIS) files clearly explain why this replicated data failed to be applied on the Target server. Also, understanding the process of Conflict Resolution is vital in determining why a transaction failed to apply to a target ER server.

Section 3.2 Ignore Conflict Resolution

The most important aspect to remember about Ignore Conflict Resolution is that it does not attempt to resolve any conflict. The source record will either be applied or discarded. As all conflicts are ignored – there is no need for shadow columns or shadow tables. The following table details how ER will react with Ignore Conflict Resolution based on the type of database operation from the source.

Key value exists at target?	Database Operation		
	Insert	Update	Delete
No	Apply Source Record	Discard Source Record	Discard Source Record
Yes	Discard Source Record	Apply Source Record	Apply Source Record

Section 3.3 Timestamp Conflict Resolution

With Timestamp Conflict Resolution, it is critical to insure the system times on all ER severs are synchronized with each other. The following table details how ER will react with Timestamp Conflict Resolution and the type of database operation from the source.

Key value exists at target?	Timestamp	Database Operation		
		Insert	Update	Delete
No	N/A	Apply record	Apply record (convert update to insert)	Apply record (insert into shadow delete table)
Yes	$T_{lastupdate} < T_{repl}$	Apply record (convert insert to update)	Apply record	Apply record
	$T_{lastupdate} > T_{repl}$	Discard Record		
	$T_{lastupdate} = T_{repl}$	Invoke stored procedure if defined as secondary conflict resolution, otherwise apply record.		

Note: $T_{lastupdate}$ refers to the column 'cdrtime' (part of the crcols columns needed for Timestamp Conflict Resolution) in the row of the target server. T_{repl} refers to the column 'cdrtime' of the replicated row from the source server.

Section 3.4 Stored Procedure Conflict Resolution

Stored Procedures used in Conflict resolution are executed only when one of the following happens:

- If not optimized, every time a conflict is detected. By default, SPL routines are not optimized.
- If optimized, when the replicated row is from the same server that last updates the target row – OR – it has a time stamp greater than or equal to that of the target server.

The execution of the Stored Procedure happens as described above whether it is the primary or secondary conflict resolution rule. IBM recommends the optimization of stored procedures used in ER to avoid performance problems.

Problems with SPL's can result from the following:

- Recent changes to SPL source
- System calls within the SPL that cause ER and the engine to hang
- Remote SQL statements within the SPL
- Saving the SPL without optimization

Section 3.5 Shadow Columns and Tables

Shadow Columns are the basis for Conflict Resolution. They contain the server id of the source of the data, and the time that this data was committed on the source.

Some common problems with Shadow Columns are the following:

- Migration to other servers for data synchronization
- In-place alter of table to add these columns

Shadow Tables are used to store a copy of data that is deleted from a server, until the acknowledgement is received from the other participants in the replicate. The main problem with Shadow Tables is that you can have so many of them - one for each replicate that uses timestamp conflict resolution. These tables have a common naming convention, which is 'cdr_deltab_*nnnnnn*', where *nnnnnn* is a serial number that is incremented with each new Shadow Table. The only way to remove a Shadow Table is to drop and recreate the replicate.

Section 3.6 ATS/RIS Log File Analysis

Chapter 9 of *IBM Informix Dynamic Server Enterprise Replication Guide Version 9.40* provides an excellent overview of the ATS and RIS files. When using the ATS and/or RIS files to help determine a problem with ER, focus on the information listed in following rows of these files:

Label	Name	Description
TXH	Transaction heading	This row contains any ER, SQL, or ISAM error information received when for transaction.
RRH	Replicated row heading	Contains the database operation
LRH	Local-row header	Notes is the local row was found in the delete table and not the target table
LRD	Local-row data	If ER encounters a severe error, the local row data is printed in hexadecimal format.

Seeing the error number in the ATS or RIS file tells you the exact problem. When an error number is not listed, look at the RRH and the LRS rows to see if this transaction failed due to the timestamp conflict resolution rule.

Section 3.7 Conflict Resolution Exercise

Using the ER servers that you stated in Section 1.6, do the following:

1. Create the following table on both ER servers:

```
CREATE TABLE rocket
  (
```



```
rocket_id int,
rocket_name     char(20),
rocket_cost     money(10,2),
launch_date     datetime year to second,
PRIMARY KEY (rocket_id, rocket_name)) WITH crcols;
```

2. Create a new replicate for this table with timestamp conflict resolution, transaction level scope, and error logging for both RIS and ATS spooling. For example:
```
cdr define repl -Ctimestamp -e 10 -S tran -R -A rep_rocket \
"test@g_edge:informix.rocket" "select * from rocket" \
"test@g_bono:informix.rocket" "select * from rocket"
```

3. Start the replicate and insert the following rows:
```
INSERT INTO rocket VALUES (10,"Gemini",500000.00,today);
INSERT INTO rocket VALUES (20,"Apollo13",800000.00,today);
INSERT INTO rocket VALUES (30,"Ramjet2",1000000.00,today);
```

4. Suspend your replicate using the 'cdr suspend replicate' command.
5. Perform the following updates on the servers specified. Perform them in the sequence indicated (T1 being the first).

(T1) on server A
```
UPDATE rocket SET rocket_cost = 600000.00
  WHERE rocket_cost = 500000.00
  AND rocket_name = "Gemini";
```

(T2) on server B
```
UPDATE rocket SET rocket_cost = 700000.00
  WHERE rocket_cost = 500000.00
  AND rocket_name = "Gemini";
```

6. Resume your replicate. For example:
```
cdr resume repl rep_rocket
```

7. Will the value of Gemini's cost be the same in both servers?
   Yes, they should have the same value because only 1 server wins the conflict.

8. What will the value be?
   It should be 700,000.00.

9. Where did conflict detection occur?
   Both servers detected a conflict. Server A received an insert for a record that already existed – so it was changed to an Update. Server B received the same type of transaction also, but had to take a different course of action to resolve the conflict.

10. Check to see if any log files were generated. There should be one of each. Answer the following questions:
    What does the ATS file contain?


```
TXH RIS file:/usr/940/ris/ris.g_bono.g_edge.D_3.030714_15:26:39.1 has
also been created for this transaction
==========
TXH Source ID:8 / Name:g_edge / CommitTime:03-07-14 15:25:02
TXH Target ID:9 / Name:g_bono / ReceiveTime:03-07-14 15:26:39
TXH Number of rows processed when transaction was aborted:1
TXH One or more rows in a transaction defined with tx scope were rejected
TXH CDR:14 (Error: Failed conflict resolution rule) / SQL:0 / ISAM:0
----------
RRH Row:1 / Replicate Id: 524297 / Table: test@informix.rocket /
DbOp:Update
RRS 8 (g_edge)|1058214302 (03/07/14 15:25:02)
RRD 10|Gemini|600000.00|2003-07-14 00:00:00
```

What does the RIS file contain?

```
TXH Source ID:8 / Name:g_edge / CommitTime:03-07-14 15:25:02
TXH Target ID:9 / Name:g_bono / ReceiveTime:03-07-14 15:26:39
----------
RRH Row:1 / Replicate Id: 524297 / Table: test@informix.rocket /
DbOp:Update
RRH CDR:14 (Error: Failed conflict resolution rule) / SQL:0 / ISAM:0
LRS 9 (g_bono)|1058214327 (03/07/14 15:25:27)
LRD 10|Gemini|700000.00|2003-07-14 00:00:00
RRS 8 (g_edge)|1058214302 (03/07/14 15:25:02)
RRD 10|Gemini|600000.00|2003-07-14 00:00:00
==========
TXH Transaction aborted
TXH ATS file:/usr/940/ats/ats.g_bono.g_edge.D_3.030714_15:26:39.2 has
also been created for this transaction
```

Cross Reference with the user manual to determine what the codes in these files represent. (TXH, RRH, RRS, RRD, LRH, LRS, LRD).

Section 4 Data Synchronization Problems

Section 4.1 Introduction

Engines and ER servers can run into problems which cause interruption of replicated data. As a result, you need to be able to determine your data is synchronized across all ER servers. This section provides ideas and tools to help you stay synchronized.

Section 4.2 Determining current status of synchronization

Several methods can be used to determine if your data is synchronized. The following table describes some of these methods:

Method	Description
Select count(*)	A simple way to check if the row counts are the same. This method will not tell you about rows that are different. Consider setting the isolation level to 'dirty read' before running this on a live table.
Select from (site A) where PK not in (site B)	This method will check the Primary Key between sites. However, it will not check for data outside the PK that may have been updated.
Select from (site A) where PK not in (site B) and A.crcols <> B.crcols	This method checks the PK and checks for updates to other row data by comparing the shadow columns. However, this method will only work with replicates with timestamp conflict resolution, and may not work well in ER environments with more than 2 servers.
Unload to files, then use diff to compare	This is a 'manual' operation and can be very time consuming. Consider only as a last resort.

If ER has produced several ATS and/or RIS files as a result of a table being locked, or maxed out of extents, consider your table(s) out-of-sync.

Section 4.3 Methods of synchronization

The amount of data that is not synchronized will be a determining factor in which method you choose for synchronization. The following table describes some of these methods:

Method	Description
High Performance Loader	This is the fastest option for large data sets, but requires a good deal of setup before running. Use 'Express' mode for data loads as it will load the CRCOLS data.
Dbimport/dbexport	Not a good option for ER. They require exclusive access to the database and cannot be used while ER is active.
Dbload	A good choice if you have a set of data that needs to be replicated to other sites. Data loaded into active replicates

	will be replicated.
Onunload/onload	Not a good fit with ER active. Both utilities require database logging to be off. The table or database loaded cannot exist, as the utility creates it. Lastly, when tables are loaded, they do not retain any information on constraints, triggers, and default values.
Unload/Load	Data loaded into active replicates will be replicated unless the transaction is started with the following syntax: `BEGIN WORK WITHOUT REPLICATION`.
Select from (site A) where PK not in (site B)	If you have a small set of data that needs to be synced, this may be a good option. There are issues with logging (addressed in Section 5.2) which you should be aware of before using this method.
Update col1 = col1	If your replicate uses timestamp conflict resolution, this option will update existing any rows, and insert any missing rows.
Drop ER, Backup, Restore, Start ER	This option requires an ER outage, but if you have a large set of data and/or replicates – this option insures you will be in sync.

Section 5 Performance Problems

Section 5.1 Introduction

What do you do when your data is replicating …slowly? Usually, these problems can be attributed to: a large replicated transaction, a network connection failure, or some resource intensive process. Also, most ER performance problems turn out to be the result of a configuration problem on the engine or network, and not ER related.

In this section you will see areas to focus on to determine which component of the engine is causing the problem.

Section 5.2 Engine Related Components

Engine related components include any environment variables, configuration parameters, and database operations. Problems with these components are not usually obvious, so some investigation is required to eliminate them as a cause of the problem. The following is a list of items to check and correct as needed:

- **Database Logging Method**: IBM Informix recommends use of unbuffered databases wherever possible. Buffered logging evaluates log records in batches and consumes excess CPU and memory. Buffered logging also introduces some latency issues as replicated data is held in the log buffer waiting to be flushed.
- **Locking Mechanisms** : For any replicated table, use row level locking to minimize contention between the application and ER.
- **Logical Log Guidelines**: Use the following guidelines when configuring logical logs:
  - Keep log files the same size
  - Logs should be large enough to accommodate 15 minutes of normal activity
  - Insure you have 4 times the logs needed to hold the maximum transaction size
  - Set LTXEHWM to 30% ore than LTXHWM
  - If using logging for the CDR_QDATA_SBSPACE, be aware this will consume additional logical log space.

  Refer to the *IBM Informix Dynamic Server Enterprise Replication Guide* for the latest suggestions on configuring logical logs on your system.

Section 5.3 Network Related Components

Network related components include network configuration (DNS, NIS, subnets, network cards, etc…), and onconfig parameters. Some of the issues to examine for tuning for performance are:

- **DNS/NIS settings**: Consider using the IP address and the port number in the sqlhosts file to avoid any name resolution problems which can happen with networks using DNS/NIS servers.
- **Subnets or Segmentation**: Consider placing your ER servers on their own network segment to avoid network contention with the normal network activity.

- **Network Cards**: With high use systems, consider the addition of another network card to offload the ER traffic from other network traffic coming to the server.

Section 5.4 Replicate Related Components

Replicate related components are replicate attributes which can have an impact on the throughput of data. Some guidelines to keep in mind when reviewing replicate definitions are:

- **Replicate 'where' clause**: Replicate filters cause ER to evaluate every row, which can lead to performance issues with the Grouper function. By not using a where clause, you can minimize the evaluation phase and increase throughput.
- **Stored Procedures**: Never use SPL as a primary conflict resolution rule, use only as a secondary rule. Stored Procedures consume considerable resources. Also, confirm any replicate using SPL for conflict resolution was created with the '—optimize' option (see Appendix A of the *IBM Informix Dynamic Server Enterprise Replication Guide* for more information).

Summary

What you should know

You should now be familiar with the following types of potential problems and how to resolve them:

- Troubleshooting Configuration Issues
- Troubleshooting Replicate Issues
- Conflict Resolution Problems
- Data Synchronization Problems
- Performance Problems

Also, you should be familiar with:

- Where to look for error information (online.log, 'cdr error', ATS/RIS files, onstat)
- Ways to try and correct some of the basic problems
- Recommendations for setting up, and running in ER environments

For more information

For more information, refer to the manual *IBM Informix Dynamic Server Enterprise Replication Guide*. You can also view the manual in PDF form at http://www-3.ibm.com/software/data/informix/pubs/library/interim/ct1t2na-pdf.html.